VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Komprese emailových kolekcí

Compression of Email Collections

2011 Jan Navrátil

Prohlášení studenta

„Prohlašuji, že jsem tuto bakalářskou/diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.“

Datum: ..... Podpis: .....

Rád bych tímto poděkoval svému vedoucímu diplomové práce Ing. Janu Platošovi Ph.D. za poskytnutí odborné pomoci při zpracování tohoto tématu a také své rodině za trpělivost a podporu.

Abstrakt

V dnešní době, kdy internetové připojení vlastní již téměř každá domácnost, se emailová komunikace stává nedílnou součástí našich soukromých i pracovních životů. To ssebou ovšem ale přináší jak pro běžné uživatele, tak pro firmy nebo společnosti tyto služby nabízející, potřebu všechny tyto emailové zprávy někde uschovávat. S příchodem multimediálních příloh se tato otázka stala ještě aktuálnější. Cílem mé práce tedy bylo seznámit se s principy emailové komunikace, podrobně prostudovat strukturu emailových zpráv a vytvořit nejdříve parser těchto zpráv a následně framework pro jejich kompresi. Výsledkem mé práce je tedy aplikace tento parser a framework využívající a nabízející několik možností komprese emailových zpráv a opětovnou dekoprimaci archivů takto vytvořených.

Klíčová slova

Email, struktura, parser, komprese, hlavička, příloha, RFC 5322, .NET

Abstract

Nowadays, when almost every household owns the internet connection, e-mail communication becomes an integral part of our private and work lives. It brings, for common users and also for companies offering this services, the need to store all these e-mail messages. When people started to use multimedia attachments, this issue has became more and more important. The goal of this work was to explore the principles of e-mail communication and to study to detail the structure of e-mail messages, then create a parser of these messages and finally a framework for their compression. The result of my work is an application, which uses this parser and framework and which offers several options for compression of e-mail messages and also decompression of created archives.

Key Words

Email, structure, parser, compression, header, attachment, RFC 5322, .NET

Obsah

Seznam obrázků

Seznam tabulek

1. Úvod

Téměř každý člověk, mající přístup k internetu, dnes vlastní svou emailovou adresu. Tento způsob komunikace je velmi rozšířený, jak v soukromé, tak firemní sféře. Množství informací každodenně takto přenášených je ohromující, není tedy divu, že ti, kdo tyto emailové zprávy musí zálohovat, bedlivě hledají, jak ušetřit každé procento z velikosti takto nashromážděných dat. Ať už to je běžný uživatel nebo firma takovéto služby poskytující, pro každého představuje ušetřený prostor určitou výhodu. Někde menší, někde jdoucí do tisíců a tisíců ročně ušetřených za elektřinu spotřebovanou na provoz diskových polí. Tento fakt je tedy hlavní motivací této práce.

Mým cílem tedy bylo napsat framework pro kompresi emailových kolekcí. Základem tohoto frameworku je parser pro emailové zprávy, který se při kompresi používá. Hlavním úkolem bylo nalézt nejvhodnější způsob zpracování jednotlivých částí emailových zpráv. Výsledkem mé práce je tedy aplikace napsaná v jazyce C#, která tento framework využívá a nabízí několik způsobů komprese emailových zpráv. Od těch bezztrátových majících účinnost okolo 60% až po ty ztrátové, zachovávající pouze nejdůležitější informace, jejichž účinnost je ale opravdu velmi lákavá.

V následující kapitole se blíže seznámíme se standarty popisujícími emailovou komunikaci. Podíváme se na to, jaké protokoly se zde používají. A kde najdeme dokumenty, které tyto standarty (nebo doporučení) obsahují.

Ve třetí kapitole se nachází popis struktury emailové zprávy, odkazy na RFC, které se touto tématikou zabývají. Plus stručný pohled na stavbu hlavičky, těla a příloh emailových zpráv.

Další kapitola už obsahuje můj návrh zpracování jednotlivých částí emalové zprávy, přičemž největší část je věnována zpracování položek hlavičky.

Následuje kapitola zabývající se požadavky na kompresi emailových zpráv, potom zde je kapitola o implementaci a na závěr pak kapitola hodnotící výsledky mé práce.

2. Standardy popisující emailovou komunikaci

2.1 Stručný popis emailové komunikace

2.1.1 Komunikační protokoly

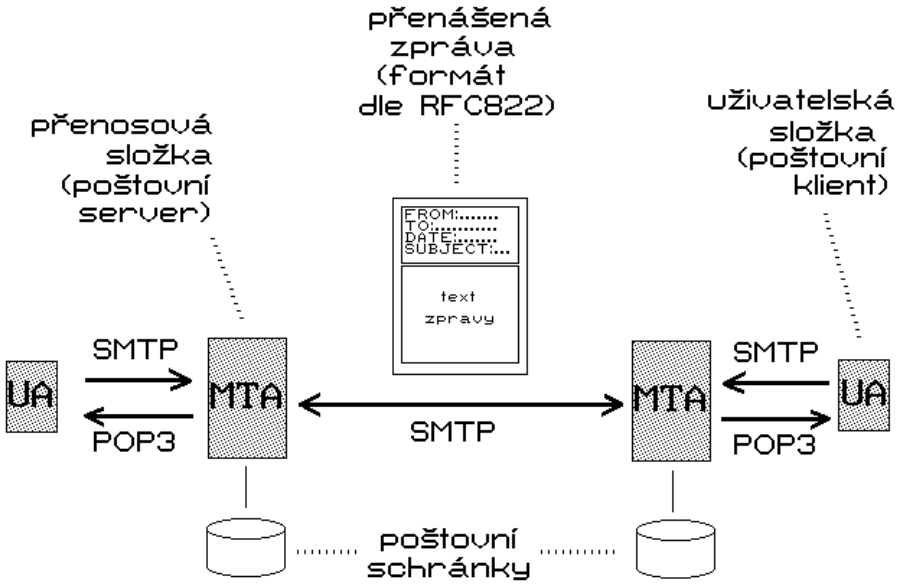
Emailové zprávy se v prostředí internetu vyměňují mezi počítači pomocí *Simple Mail Transfer Protocol* a pomocí softwaru typu *Mail Transfer Agent*. Uživatelé mají na svém počítači nainstalován program, který se nazývá emailový klient. Ten slouží ke stahování zprávy z poštovního serveru, nejčastěji za použití protokolů *POP* nebo *IMAP*.

Příjmuté emaily je možné ukládat buď na straně serveru, nebo na straně klienta. Standardní formáty pro takovéto emailové schránky jsou např. *Maildir* nebo *mbox*. Několik emailových klientů používá vlastní formát a na konverzaci mezi těmito formáty je potom potřebný speciální program.

Někteří uživatelé nepoužívají e-mailového klienta, ale přistupují ke zprávám umístěným na poštovním serveru přes webové rozhraní. Tento postup se často používá zejména u freemailových (bezplatných) služeb.

2.1.2 Kódování obsahu e-mailu

Pro emailové zprávy je definován přenos 7bitové *ASCII* informace. Přesto je většina emailových přenosů 8bitových, kde ale již nelze zaručit bezproblémovost. Z toho důvodu byla elektronická pošta rozšířena o standard *MIME*, což umožnilo kódování vkládaných *HTML* nebo binárních příloh, obrázků, zvuků a videí.



Obrázek 1 - Znázornění průběhu emailové komunikace.

2.2 Popis jednotlivých částí

2.2.1 Simple Mail Transfer Protocol

SMTP je interntový protokol určený pro přenos emailových zpráv mezi dalšími přepravci elektronické pošty (MTA). Jedná se o jednu z nejstarších aplikací - původní norma *RFC 821* byla vydána v roce 1982 (v roce 2001 ji nahradila novější *RFC 2821* a tu v roce 2008 nahradila stávající norma *RFC 5321*).

Tento protokol zajišťuje doručení pošty pomocí přímého spojení mezi odesílatelem a adresátem. Emailová zpráva je doručena do tzv. poštovní schránky adresáta, ke které potom může uživatel kdykoli (off-line) přistupovat (vybírat zprávy) pomocí protokolů POP nebo IMAP. SMTP funguje nad protokolem TCP, pro svoji funkci využívá port TCP/25.

Architektura pošty

Doručování elektronické pošty po internetu se účastní tyto tři druhy programů:

- MUA - *Mail User Agent*, poštovní klient, který zpracovává zprávy u uživatele.
- MTA - *Mail Transfer Agent*, server, který se stará o doručování zpráv k adresátovi.
- MDA - *Mail Delivery Agent*, program pro lokální doručování, který umísťuje zprávy do uživatelských schránek, případně je může přímo automaticky zpracovávat (ukládat přílohy, odpovídat, spouštět různé aplikace pro zpracování apod.)

Poštovní klient

Poštovní klient je program, který zajišťuje odesílání zpráv a vybírání schránek. Mezi nejznámější patří např. *Microsoft Office Outlook* nebo *Mozilla Thunderbird*. Je to v podstatě specializovaný editor, který umí, kromě vytvoření zprávy, také manipulovat se schránkami, odeslat zprávu nejbližšímu MTA a převzít zprávu ze serveru prostřednictvím POP3 nebo IMAP. Vlastním doručením zprávy až k adresátovi se klient nezabývá.

Poštovní server

Poštovní server běží obvykle jako démon a naslouchá na portu TCP/25. K tomuto portu se může připojit (navázat TCP spojení) buď poštovní klient, nebo jiný server, který předá zprávu k doručení. MTA zkontroluje, zda je zpráva určena pro systém, na kterém běží. Pokud ano, předá ji programu MDA (lokální doručení). Pokud je zpráva určena jinému počítači, naváže spojení s příslušným serverem a zprávu mu předá.

Program pro lokální doručování

Server by mohl zprávy do uživatelských schránek ukládat přímo, ale výhodnější je k tomu použít specializovaný program. To umožňuje při doručování ještě dále zprávy zpracovávat nebo filtrovat. Příkladem může být třídění zpráv do různých schránek uživatele podle obsahu (odesilatele, subjektu apod.), nebo odstraňování nežádoucích zpráv (viry, spam). Tyto volby si může každý uživatel nastavit samostatně nezávisle na ostatních.

2.2.2 POP3

Post Office Protocol version 3 je internetový protokol, který se používá pro stahování emailových zpráv ze vzdáleného serveru na klienta. Jedná se o aplikační protokol pracující přes TCP/IP připojení. Tento protokol byl standardizován v roce 1996 v RFC 1939.

POP3 je následníkem protokolů POP1 a POP2 (označení POP už dnes téměř výhradně znamená POP3). V současné době používají téměř všichni uživatelé elektronické pošty pro stahování emailů programy využívající POP3 nebo IMAP.

Ze vzdáleného serveru se stáhnou všechny zprávy, i ty, které uživatel číst nechce, nebo třeba spam (pokud ho již nevyfiltroval poštovní server). Většina POP3 serverů sice umožňuje stáhnout pouze hlavičky zpráv (a následně vybrat zprávy, které se mají stáhnout celé), ale podpora v klientech vesměs chybí. Tuto nevýhodu může odstranit protokol IMAP, který pracuje se zprávami přímo na serveru.

2.2.3 IMAP

Internet Message Access Protocol je internetový protokol pro vzdálený přístup k emailové schránce. Na rozdíl od protokolu POP3 vyžaduje IMAP trvalé připojení (tzv. on-line), avšak nabízí pokročilé možnosti vzdálené správy (práce se složkami, přesouvání zpráv, prohledávání na straně serveru apod.) V současné době se používá protokol **IMAP4** (*IMAP version 4 revision 1* - IMAP4rev1), který je definován v RFC 3501.

Jak jsme siž řekli, protokol IMAP vyžaduje trvalé připojení k emailové schránce. Díky tomu je možné s celou poštovní schránkou plně pracovat z libovolného místa. Všechny zprávy a složky jsou uloženy na poštovním serveru a na počítač se stahují jen nezbytné informace. Takže při zobrazení složky se stáhnou jen záhlaví zpráv a jejich obsah až v případě, že zprávu chce uživatel přečíst. U jednotlivých zpráv se uchovává jejich stav (nepřečtená, odpovězená, důležitá), uživatel může zprávy přesouvat mezi složkami, složky vytvářet, mazat, prohledávat na straně serveru apod. Protokol umožňuje současné připojení více klientů.

2.2.4 MIME

MIME, plným názvem *Multipurpose Internet Mail Extensions* („Víceúčelová rozšíření internetové pošty“), je internetový standard, který umožňuje pomocí elektronické pošty zasílat zprávy obsahující text s diakritikou, přidávat ke zprávám přílohy v nejrůznějších formátech, nebo umožňuje např. funkci

digitálního podpisu apod. V současné době ho využívají i další protokoly aplikace (např. HTTP). Standard MIME je definován šesti dokumenty: RFC 2045, RFC 2046, RFC 2047, RFC 4288, RFC 4289 a RFC 2049.

Původní standard elektronické pošty byl vytvořen tak, aby umožňoval přenos anglického textu, k čemuž stačí pouze tabulky znaků ASCII. Proto nebylo dlouho možné používat v elektronické poště znaky s diakritikou a posílat současně se zprávou i přílohy. Částečným řešením bylo například použití *uuencodingu* nebo jiných metod, avšak citelně scházela celosvětová standardizace.

MIME rozšiřuje formát emailových zpráv o tyto možnosti:

- podpora textu psaného ve znakových sadách jiných než US-ASCII
- podpora příloh (obrázky, zvuky, filmy, programy a podobně)
- vícedílné zprávy
- informace v hlavičce v jiné znakové sadě než ASCII

Základní formát emailové zprávy je definován v RFC 5322. Tento standard specifikuje formátování hlaviček, těla e-mailu a pravidla pro běžně používané pole hlavičky jako „Komu:“, „Předmět:“, „Od:“ a „Datum:“. MIME definuje sadu hlaviček pro specifikaci doplňkových atributů zprávy obsahující "content-type" a definuje sadu "transfer-encoding", která může být použita pro reprezentování 8bitových binárních dat užívajících znaky 7bitového ASCII.

Na závěr této kapitoly bych chtěl uvést, že většina informací zde použitých byla čerpána především ze zdrojů [4] a [5].

3. Přehledný popis struktury emailové zprávy

Co nás v této práci bude zajímat nejvíce, jsou samotné emailové zprávy a jejich komprimace. Proto se v této kapitole podrobně zaměříme na strukturu takové zprávy. Formát zpráv elektronické pošty specifikuje RFC 5322, jenž nahrazuje starší RFC 2822 a 822.

Nejprve si připomeňme samotný statut dokumentu označovaného jako RFC5322 - jak jsme si již uvedli v předchozí kapitole, dokumenty RFC (Request For Comment) jsou základními technickými dokumenty, které se zabývají nejrůznějšími aspekty fungování Internetu, včetně protokolů TCP/IP. Nemají formu právně závazných norem (proto je namístě označovat je spíše pouze jako doporučení). Ovšem některé z nich - zvláště ty, které definují různé protokoly - však mají povahu skutečných a závazných standardů, protože jak v rámci Internetu, tak i mimo něj (a to i v komerční sféře) jsou všeobecně uznávány, respektovány a důsledně dodržovány. Dokument RFC5322 je právě jedním z takovýchto dokumentů, které mají "sílu" závazného standardu.

3.1 Co popisuje dokument RFC5322?

Při přenosu emailových zpráv mezi jednotlivými přenosovými složkami je třeba uvažovat jak konkrétní mechanismus, kterým si tyto složky zprávy předávají, tak i samotný vnitřní formát těchto zpráv (kterému zmíněné přenosové složky musí rozumět alespoň do té míry, aby dokázaly poznat, komu a kam mají zprávy poslat).

Každá emailová zpráva se tedy skládá jak z vlastního obsahu zprávy (tzv. těla), tak i její hlavičky, určující mj. odesilatele a konečného příjemce a obsahující i další údaje, které zde budou dále popsány. Všechny tyto části vyplňuje tzv. uživatelská složka, resp. program, jehož prostřednictvím uživatel své zprávy sestavuje. Když tato uživatelská složka (User Agent) předá připravenou zprávu přenosové složce ke skutečnému odeslání, ta následně přidá takové údaje, jaké jsou zapotřebí pro přenos zprávy (načež zajistí její skutečný přenos). Při sestavování těchto údajů přitom přenosová složka vychází z toho, co je napsáno v hlavičce zprávy (konkrétně z údajů o odesilateli a příjemci atd.). Sama proto musí rozumět formátu, podle kterého je zpráva sestavena. A právě tento formát je předmětem doporučení RFC5322, zatímco údaje potřebné pro přenos a její vlastní přenos se řídí protokolem SMTP, jak jsme si již popsali dříve.

3.2 Hlavička a tělo zprávy

Doporučení RFC5322 chápe zprávu jako text členěný na jednotlivé řádky a rozděluje ji do dvou základních částí: na hlavičku (header) a tělo (body). Dále se podrobněji zabývá syntaxí a sémantikou hlavičky, zatímco obsah těla zprávy nijak přesněji nevymezuje - pouze říká, že hlavička musí předcházet tělu a tělo musí být od hlavičky odděleno nejméně jednou prázdnou řádkou (neboli: vše, co následuje za první prázdnou řádkou, je považováno za tělo zprávy). Podrobněji se na tělo zprávy podíváme v jedné z následujících podkapitol.

Na hlavičku zprávy se doporučení RFC5322 dívá jako na posloupnost položek, kterým v originále říká header fields (doslova: pole hlavičky). Specifikují metainformace k dané zprávě. Každá položka musí začínat na novém řádku (přesně na první pozici řádku) a může pokračovat i na dalších řádcích (v takovém případě ale nikoli od první pozice - každý další řádek začíná ' ' nebo '\t').

Každá položka je vždy uvozena určitým klíčovým slovem (zakončeným dvojtečkou), které definuje její význam (a tím současně ulehčuje práci programům, jako je např. i ten náš, které mají hlavičku analyzovat a řídit se jejím obsahem). Za tímto uvozujícím klíčovým slovem (a povinnou dvojtečkou) pak následuje vlastní obsah příslušné položky.

Některým položkám hlavičky doporučení RFC5322 předepisuje určitou povinnou syntaxi, zatímco jiným nikoli. Logika je zde taková, že obsah některých položek analyzují programy, které se na přenosu zpráv podílí, a povinná syntaxe jim v tom velmi účinně pomáhá. Týká se to především položek, které obsahují adresy a údaje o datu a čase. Naproti tomu u jiných položek (například u položky Subject, vyjadřující předmět zprávy) prakticky nepřipadá v úvahu, že by je bylo potřeba analyzovat, a tak jejich syntaxe není nijak předepsána - mohou to tedy být libovolné texty.

Doporučení RFC5322 také nepředepisuje povinné pořadí jednotlivých položek hlavičky (pouze vyslovuje určitá doporučení). Zde je příklad hlavičky sestavené podle RFC5322:

```
Received: from mail-iw0-f182.google.com (mail-iw0-f182.google.com
      [209.85.214.182])by email-mx8.go.seznam.cz (Seznam SMTPD
      1.2.15-3@16648M) with ESMTP;
      Sun, 27 Feb 2011 14:00:46 +0100 (CET)
Received: by iwn33 with SMTP id 33so2849697iwn.41
      for <honza.n@seznam.cz>; Sun, 27 Feb 2011 05:00:43 -0800 (PST)
Date: Sun, 27 Feb 2011 14:00:43 +0100 (CET)
From: =?us-ascii?Q?Jan=20Navratil?= <honza.n@gmail.com>
Message-Id: <AANLkTinWKfBb1UfnJQJWsM5ZT_JS-syuCqwbPf=ty-
      BW@mail.gmail.com>
To: Honza <honza.n@seznam.cz>
Subject: Jak je ve Francii?
```

3.3 Položky hlavičky

Samotné doporučení RFC5322 definuje poměrně velký počet různých položek pro hlavičky emailových zpráv. Cílem této kapitoly tedy samozřejmě nebude podat jejich kompletní seznam, nebo dokonce popis - od toho je samotný dokument RFC5322, který je určen právě pro tyto účely a je volně dostupný. My si zde uvedeme přehled většiny nejdůležitějších položek a ukážeme si, jaké možnosti jejich použití nabízí dnešní systémy elektronické pošty.

Každá zpráva samozřejmě musí od někoho pocházet. Doporučení RFC5322 uvádí autora zprávy v položce "**From**", například:

```
From: nav220@vsb.cz
```

Tato položka je přitom jednou z těch, u kterých je předepsána povinná syntaxe. Ta však stále nabízí řadu variant, včetně možností vkládání komentářů, srozumitelných člověku. Například:

```
From: nav220@vsb.cz (Jan Navrátil)
```

apod. Doporučení RFC5322 dále pamatuje i na možnost, že původní autor zprávy není totožný s jejím skutečným odesilatelem. K takovéto situaci může dojít například tehdy, když někdo posílá vzkaz někoho jiného, když si příliš zaneprázdněný šéf nechává odesílat zprávy svou sekterářkou. RFC5322 se s tím vyrovnává zavedením položky "**Sender**", která specifikuje odesilatele zprávy (je-li to někdo jiný než autor zprávy, pro kterého je určena položka "From").

Další položkou, která souvisí s možností, že autorem zprávy je někdo jiný než odesilatel, je položka "**Reply-To**". Ta totiž explicitně říká, kam má být zaslána odpověď. Pomocí položky "Reply-To" lze explicitně předepsat, kam mají být odpovědi posílány. Je to užitečné například i pro uživatele, kteří mají více pracovišť, resp. svou elektronickou poštu odesílají z více různých míst - pomocí této položky si pak mohou nastavit, aby jim odpovědi chodili na jedno konkrétní místo.

Podobnou syntaxi jako položka "From" má i položka "**To**", určující adresáta zprávy. Zprávy elektronické pošty je možné zasílat také i jako tzv. kopie na vědomí (Carbon Copy), a dokonce i jako tzv. "slepé" kopie (Blind Carbon Copy) - o kterých se hlavní adresát zprávy nedozví (na rozdíl od příjemců běžných kopií, o kterých se v hlavičce svého originálu dozví). Pro specifikaci adresátů kopií jsou tedy určeny položky "**Cc**" a "**Bcc**".

V každé přijaté zprávě bývá obvykle obsažen určitý počet položek "**Received**", které obsahují záznamy o "cestě" zprávy mezi jednotlivými přenosovými složkami - v zásadě by bylo možné říci, že každému jednotlivému "přeskoku" by měla odpovídat jedna tato položka (a jedna další obvykle připadá i na první přenosovou složku, která zprávu přijme k odeslání). V položkách "Received" pak

může být vyjádřeno například i to, jakým protokolem byla zpráva přenesena, pod jakým označením apod. - tyto informace jsou většinou určeny pro potřeby ladění a řešení nestandardních situací. Například:

```
Received: from mail-iw0-f182.google.com (mail-iw0-f182.google.com
[209.85.214.182]) by email-mx8.go.seznam.cz (Seznam SMTPD 1.2.15-3@16648M)
with ESMTP; Sun, 27 Feb 2011 14:00:46 +0100 (CET) Received: by iwn33 with
SMTP id 33so2849697iwn.41 for <honza.n@seznam.cz>; Sun, 27 Feb 2011 05:00:43
-0800 (PST) Received: by 10.231.179.230 with SMTP id
br38mr1957826ibb.103.1298811643384; Sun, 27 Feb 2011 05:00:43 -0800 (PST)
Received: by 10.231.51.87 with HTTP; Sun, 27 Feb 2011 05:00:43 -0800 (PST)
```

V RFC5322 je zmíněna ještě jedna položka sloužící k trasování, a to "**Return-Path**".

Další položky, na které RFC5322 pamatuje, umožňují vyjádřit datum a čas, kdy byla zpráva zadána k odeslání (povinná položka "**Date**"), její předmět (položka "**Subject**") apod. Položku Date nebo např. identifikační položku "**Message-Id**", která zabraňuje vícenásobnému doručení zprávy, generuje většinou emailový klient automaticky. Stejně tak generuje i položku "**Content-type**" obsahující informace o tom, jak má být zpráva v klientu zobrazena.

Mezi identifikační položky dále patří "**In-Reply-To**" a "**References**". Mezi informační položky potom "**Comments**" a "**Keywords**".

Ještě zde můžeme uvést skupinu položek týkajících se přeposílání zprávy, jsou to tyto položky:"**Resent-Date**", "**Resent-From**", "**Resent-Sender**", "**Resent-To**", "**Resent-Cc**", "**Resent-Bcc**" a "**Resent-Message-ID**".

3.4 Tělo zprávy

Stejně jako dříve u dopisů i u elektronikých zpráv tvoří hlavní část tělo - obsah, kvůli kterému celou zprávu posíláme. Tvoří jej text, pro který nejsou dána žádná specifická pravidla určující jeho strukturu, či povinné položky. Od hlavičky emailové zprávy je oddělen prázným řádkem a v některých případech bývá zakončen elektornickým podpisem.

Tělo samotné je tvořeno ASCII textem, ale může obsahovat i ne-ASCII znaky, pokud jsou uvedeny MIME hlavičkou Content-Transfer-Encoding: 8bit (tím se ale již zabývají další RFC dokumenty). Ještě platí jedno pravidlo, a to, že jeden řádek těla zprávy nesmí obsahovat více jak 998 znaků a neměl by se skládat z více jak 78.

3.4.1 Kódování obsahu

Elektronická pošta byla původně navržena pro 7-bitové ASCII kódování. Většina programů dnes používaných pro práci s emaily nemé problém s 8-bitovými znakovými sadami, ale přesto stále musí předpokládat, že můžou komunikovat se 7-bitovými servery nebo emailovými klienty. Standart MIME nám přinesl charakteristiku používané znakové sady a dva typy kódování obsahu pro umožnění přenosu dat, která není možné převést do 7-bitového ASCII. Jsou to tatko kódování: quoted printable pro většinu 7-bitového obsahu a pro pár znaků mimo tento rozsah a potom base64 pro libovolná binární data. Aby bylo možné přenášet emailové zprávy i bez těchto dvou kódování byli představeny dvě rozšíření 8BITMIME a BINARY.

3.4.2 Plain text a HTML

U většiny moderních grafických emailových klientů si uživatel může zvolit, zda pro tělo zprávy použije formát plain text nebo HTML. Pokud uživatel zvolí formát HTML, emailový klient často zároveň z důvodu kompatibylity automaticky vygeneruje kopii v plain textu pro starší klienty, kteří si s HTML nedokáží poradit.

Mezi hlavní výhody HTML patří možnost vložit do zprávy odkazy nebo obrázky, vlkádat citace předchozích zpráv, přirozené zarovnání zprávy na jekémkoliv displeji, použití tučného písma nebo kurzívy anebo změnit font písma. Mezi nevýhody se řadí především větší velikost takovýchto emailových zpráv nebo možná hrozba zneužití HTML k phishingovému útoku nebo vložení jiného srytého škodlivého softwaru.

Některé weby přímo doporučují pro své emailové konference z výše uvedených důvodů odesílat všechny příspěvky v plain textu. Je to ale i proto, že podstatná část jejich čtenářů používá textové emailové klienty, kteří HTML nepodporují.

Některé emailové klienty od Microsoftu nabízejí bohaté možnosti formátování za pomocí RTF. Toho by se ale uživatelé měli vyvarovat, pokud nemají jistotu, že příjemce má kompatibylní emailový klient.

Na závěr je ještě potřeba zmínit, že pokud HTML používáte, tak je tuto skutečnost nutné uvést také v hlavičce emailové zprávy. Konkrétně se jedná o položku "Content-type: text/html". To za vás ale v drtivé většině případů zajistí program starající se o odeslání emailu.

3.5 Příloha zprávy

Pokud emailová zpráva obsahuje přílohu nebo dokonce více příloh, ty následují za tělem emailu a jsou uvedeny krátkou hlavičkou. Toto téma spadá pod standart MIME, o kterém jsme se již zmiňovali dříve. MIME definuje vlastní sadu hlaviček pro specifikaci doplňkových atributů zprávy obsahující "content-type" a definuje sadu "transfer-encoding", která může být použita pro reprezentování 8bitových binárních dat užívajících znaky 7bitového ASCII.

3.5.1 Položky MIME hlavičky

MIME-Version

Přítomnost této hlavičky značí, že je zpráva formátována MIME. Typická hodnota je „1.0“:

Content-Type

Tato hlavička označuje typ media (text, audio, video,…) v těle zprávy. Skládá se z typu a podtypu a popřípadě doplňkové informace uvedené za středníkem (parametr). Informuje příjemce o obsahu zprávy.

- **Typ** - definuje o jaký typ souboru se jedná (text, obrázek, video, zvuk,…)
- **Podtyp** - definuje formát souboru
- **Doplňkové informace** - např. parametr udávající hodnotu

Content-Disposition

Původní specifikace MIME popisovala pouze strukturu emailové zprávy, tím, jak zprávu zobrazit se vůbec nezabývala. V RFC2183 tedy byly nově specifikovány různé prezentační styly zprávy a byla přidána hlavička Content-Dissposition. Nová MIME část tedy může obsahovat:

- *Inline* content-disposition, což znamená, že tato hlavička bude zobrazena zároveň při zobrazení zprávy.
- *Attachment* content-disposition, v tomto případě se nezobrazí automaticky, ale je za potřebí aby ji zobrazil sám uživatel svou akcí.

Navíc k této informaci hlavička content-disposition také poskytuje informace o názvu souboru a datu jeho vytvoření nebo modifikace.

Následující příklad je přímo z RFC2183:

```
Content-Disposition: attachment; filename=genome.jpeg;
        modification-date="Wed, 12 February 1997 16:29:51 -0500";
```

Content-Transfer-Encoding

Definuje sadu metod pro reprezentaci binárních dat v textovém formátu ASCII.

- *7bit* - krátké řádky, pouze us-ASCII znaky
- *quoted-printable* – pro us-ASCII texty obsahující malé procento ne-ASCII znaků
- *base64* – pro 8bitová netextová data a texty skládající se z většiny ne-ASCII znaků
- *8bit* - krátké řádky, možnost ne-ASCII znaků
- *binary* - dlouhé řádky, možnost ne-ASCII znaků

Content-ID

Označuje identifikátor v hlavičce multi-part zpráv pro jednoznačnou identifikaci dané části zprávy. Slouží k vytvoření odkazu z jedné části zprávy na druhou.

Syntaxe Encoded-word

Počínaje RFC2822 jsou hlavičky a jejich hodnoty v ASCII znacích, pokud chceme použít jinou znakovou sadu, musíme použít "Encoded-word" syntaxi. V syntaxi je zahrnut jak originální text v ASCII, tak požadovaná cílová znaková sada a dekodovací metoda („content-transfer-encoding“), pomocí těchto parametrů se pak text zobrazí ve správné znakové sadě.

Forma zápisu je: "=?*charset*?*encoding*?*encoded text*?=".

- *charset* může být jakákoliv znaková sada registrovaná u IANA. Většinou se jedná o stejnou znakovou sadu, jakou používá tělo zprávy.
- *encoding* může být "Q" značící Q-encoding, nebo "B" značící base64.
- *encoded text* je text zakódovaný buď pomocí Q-encoding nebo base64.

Například: `Subject: =?iso-8859-1?Q?=A1Hola,_se=F1or!?=`

interpretujeme jako "*Subject: ¡Hola, señor!*".

3.5.2 Vícedílné zprávy

Vícedílné zprávy (MIME multipart) obsahují v hlavičce „content-type“ oddělovač (boundary). Oddělovač je umístěn mezi částmi zprávy + na začátek a na konec těla zprávy. Zde je příklad:

```
MIME-version: 1.0
Content-type: multipart/mixed; boundary="frontier"

--frontier

Content-type: text/plain

--frontier

Content-type: application/octet-stream
Content-transfer-encoding: base64

PGh0bWw+CiAgPGhlYWQ+CiAgPC9oZWFkPgogIDxib2R5PgogICAgPHA+VGhpcyBpcyB0aGUg
Ym9keSBvZiB0aGUgbWVzc2FnZS48L3A+CiAgPC9ib2R5Pgo8L2h0bWw+Cg==
--frontier--
```

Každá část se skládá ze své vlastní hlavičky a těla. Multipart bloky jako celek nemají znakovou sadu, je zde třeba použít syntaxi "encoded-word", pokud nepracujeme s ASCII, těla jednotlivých částí zprávy mohou použít znakovou sadu náležící do jejich "content-typu".

Podtypy vícedílných zpráv

MIME standard definuje různé podtypy vícedílných zpráv (multipart), které specifikují druh jednotlivých částí zpráv. Podtyp je definován v „content-type“ hlavičce celé zprávy. Například multipart zpráva používající podtyp „digest“ by byla v hlavičce zapsána jako „multipart/digest“.

Standardně používané jsou podtypy mixed a digest, ostatní jsou volitelné.

Seznam nejvíce používaných subtypů:

Mixed

„Multipart/mixed“ je používáno pro odesílání souborů s různými hlavičkami. Používá se pokud jsou jednotlivé části těla zprávy odlišné a je třeba je uspořádat. Pokud se odesílají např. obrázky, vetšina e-mailových klientů zobrazí tyto hlavičky jako vkládané (inline).

Message

Message/rfc822 obsahuje hlavička v každé zprávě.

Alternative

Podtyp „multipart/alternative“ znamená, že každá část je alternativní verze stejného (nebo podobného) obsahu, každá v jiném formátu označená svou hlavičkou. Systém si může vybrat nejlepší reprezentaci, která je nejvhodnější pro zpracování zprávy. Běžně je „multipart/alternative“ užíván pro e-mail složený ze dvou částí, jedna je prostý text (text/plain) a druhá HTML (text/html). Část s prostým textem zajišťuje zpětnou kompatibilitu, zatímco HTML část formátování a vytvoření odkazů. Většina e-mailových klientů nabízí volbu, zda prostý text preferovat před HTML, toto je příklad toho jak lokální faktory mohou ovlivnit „počínání“ aplikace, která část zprávy je „nejlepší“ pro zobrazení. Zde je, pro ilustraci, příklad:

```
From: Jan Novák <jan.novak@seznam.cz>
To: Alois Starý <a.stary@ibm.com>
Subject: Formatted text mail
MIME-Version: 1.0
Content-Type: multipart/alternative; boundary=boundary42

--boundary42

Content-Type: text/plain; charset=us-ascii

--boundary42

Content-Type: text/x-whatever

--boundary42

Content-Type: text/richtext

--boundary42--
```

Systém uživatele si v tomto příkladu nejvíce "rozumí" s formátem text/richtext, který je první v pořadí relevantnosti (poslední vypisovaný formát je nejlepší) a proto jej využije, jiný systém si však může vybrat i jeden z ostatních dvou formátů.

Signed

Podtyp „multipart/signed“ je využíván k přiložení digitálního podpisu do zprávy, která má dvě části. V první je tělo zprávy opatřené digitálním podpisem, druhá část obsahuje kontrolu digitálního podpisu.

Encrypted

Podtyp „multipart/encrypted“ slouží k šifrování zprávy a má dvě části. První obsahuje informace potřebné k dekódování, druhá část je pak určena přímo pro část určenou k dekódování. Nejběžnějšími typy jsou „application/pgp-encrypted“ a „application/pkcd7-mime“.

Zde je, pro představu, ukázka emailové zprávy obsahující drobnou přílohu ve formátu .txt. Zpráva, která by měla přiloženou např. jedinou fotku ve formátu .jpg, by zde takto zabrala minimálně 10 stránek. Hlavička přílohy je zvýrazněna tučným písmem:

```
Reply-To: =?us-ascii?Q?Jan=20Navratil?= <navratil.j@gmail.com>
Received: from mail-fx0-f48.google.com (mail-fx0-f48.google.com [209.85.161.48])
    by email-smtpd3.ng.seznam.cz (Seznam SMTPD 1.2.15-6@18565) with ESMTP;
    Thu, 28 Jul 2011 18:53:27 +0200 (CEST)
Received: by fxg7 with SMTP id 7so2035924fxg.35
    for <vesan@seznam.cz>; Thu, 28 Jul 2011 09:53:25 -0700 (PDT)
Dkim-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed; d=gmail.com; s=gamma;
    h=mime-version:date:message-id:subject:from:to:content-type;
    bh=P5ZRzpYSM9n3SBNBIWimf2mn+zS2S2stTpAQJ0oeUIs=;
    BhVFdNDpSFdYYKgsgvI+0t+cszYqWUJWpRAlG4+GpypXWswHPrrWzUgT5QrFAKoTPuIw
    ZMBl3YbsV33W9AcH85y0pBkxgjSKg+emNJXnk=
Mime-Version: 1.0
Received: by 10.223.29.10 with SMTP id o10mr284171fac.87.1311872005109;
    Thu, 28 Jul 2011 09:53:25 -0700 (PDT)
Received: by 10.223.86.72 with HTTP; Thu, 28 Jul 2011 09:53:24 -0700 (PDT)
Date: Thu, 28 Jul 2011 18:53:24 +0200 (CEST)
Message-Id: <CAPAj6N+ONPAoosAkF2y8eYGG=svK_GMA85QWkMuX1cw06otQtA@mail.gmail.com>
Subject: Hallo!
From: =?us-ascii?Q?Jan=20Navratil?= <navratil.j@gmail.com>
To: vesan <vesan@seznam.cz>
Content-Type: multipart/mixed; boundary=00151747570adaf87904a924000e
X-Smtpd: 1.2.15-6@18565
X-Session: 55
X-Country: US
X-Spam-Status:score=0.0
X-Seznam-Ffp: 1015921338

--00151747570adaf87904a924000e
Content-Type: multipart/alternative; boundary=00151747570adaf87404a924000c

--00151747570adaf87404a924000c
Content-Type: text/plain; charset=ISO-8859-1

--
JN

--00151747570adaf87404a924000c
Content-Type: text/html; charset=ISO-8859-1

<br clear="all"><br>-- <br>JN<br>

--00151747570adaf87404a924000c--

--00151747570adaf87904a924000e
Content-Type: text/plain; charset=windows-1250; name="internet.txt"
Content-Disposition: attachment; filename="internet.txt"
Content-Transfer-Encoding: base64
X-Attachment-Id: f_gqnypasb3

IkludGVybmV0IGplIHpyY2FkbGVtIHNwb2xl6G5vc3RpLiBLZHmeIHNlIHbhbSBuZWztYu0gdG8s
IGNvIHZpZO10ZSB2IHpyY2FkbGUsIG5lbu0gdG8gY2h5YmEgenJjYWRsYSwiG5hcHNhbCBWaW50
b24gQ2VyZiwgamVkZW4geiBvdGP5IFRDUC9JUCBwcm90b2tvbHUu
--00151747570adaf87904a924000e--
```

Většina informací použitých v této kapitole byla čerpána především ze zdrojů [1], [2] a [4].

4. Návrh kódování pro jednotlivé části emailu

Potom, co jsme se podrobně seznámili se strukturou emailových zpráv, můžeme přistoupit k hlavní části této práce, a to k jejich kompresi. Ve své práci jsem se nejvíce a nejdetailněji zabýval zpracováním hlavičky emailových zpráv, začnu tedy tímto. Komprimování těla zprávy, které se skládá z převážné většiny pouze z plain textu, není hlavní náplní této práce. Nejrůznějších algoritmů pro kompresi textu již bylo nalezeno více než dost a proto volbu toho optimálního přenechám na sofistikovanějších programech, než je ten můj. Stejně tak komprese multimediálních příloh, které v dnešní době můžou obsahovat soubory jakýkoliv typů a formátů. Může to být od běžných textových dokumentů, přes nejrůznější formáty fotografií, videa nebo hudby opravdu cokoliv. Popis a implementace individuálního kódování pro každý z těchto formátů by vydal za další samostatnou práci. Neznamená to ovšem, že bysme se komprimování příloh zcela vyhnuli, pouze k této problematice budeme přistupovat jednotně. Konkrétní postup si popíšeme v závěrečné části této kapitoly.

4.1 Zpracování hlaviči

Po přečtení minulé kapitoly, popř. Po prostudování příslušných RFC dokumentů, víme, že hlavička emailové zprávy se skládá z velkého množství různých polí. Některá pole jsou povinná, jiná ne. Některá vyplňuje autor zprávy sám, některá jsou generována automaticky.

Mou první myšlenkou, při vymýšlení způsobu komprese hlavičky, bylo tedy toto velké množství polí zredukovat na minimum. Vycházel jsem z předpokladu, že koncový uživatel si bude staré emailové zprávy archivovat především kvůli jejich obsahu a ne aby dlouhé zimní večery trávil procházením položek *Receive:* aby zjistil, jak „strastiplnou“ cestu zpráva musela vykonat, než k němu dorazila, nebo jaký byl její *Spam-Status*.

Z množiny veškerých polí hlavičky jsem tedy vybral následující podmožinu těch, podle mého názoru, uživatelsky nejdůležitějších:

- **From:** - od koho zpráva pochází
- **To:** - komu je zpráva určena
- **Date:** - kdy byla odeslána
- **Subject:** - jaký byl její předmět

A následně jsem do svého programu zaimplementoval možnost tzv. ztrátové komprese, která při archivaci zprávy zachová u hlavičky pouze právě tato pole. Opět zdůrazňuji, že tato možnost je tedy opravdu vhodná spíše jen pro běžné uživatele, kteří si zprávy archivují především kvůli zachování jejich obsahu a ne pro jejich opětovné budoucí praktické použití. Tato možnost tedy rozhodně není vhodná např. pro servery poskytující emailové schránky, protože nemůžou nikdy s jistotou říci, že danný uživatel už archivované zprávy nebude používat.

Pro tyto ostatní případy je zde přichystána komprese bezeztrátová. Tento druh komprese je založen na nahrazování dlouhých řetězců jejich co nejkratší variantou, při zachování původního obsahu. Zde je výčet těchto substitucí:

1. **Nahrazení názvů polí hlavičky**
2. **Nahrazení názvů domén v seznamu adresátů**
3. **Nahrazení datumů**
4. **Nahrazení „boundary“**

Ad 1.

Názvy polí hlavičky jsem nahradil jejich kratšími ekvivalenty skládajicíma se ze znaku uvozujícího zkratku a většinou z počátečních písmen názvu pro lepší přehlednost. Např. položka „Received:“ bude nahrazena zkratkou #re nebo „Date:“ nahradí jenom #d. Zde uvádím kompletní seznam použitých zkratek (hlavičky týkající se přílohy jsou uvedeny dále v samostatné části):

Received: → #re
Date: → #d
Message-ID: → #mi
Message-Id: → #mi
Subject: → #s
From: → #f
Content-Type: → #c
MIME-Version: → #mv
Mime-Version: → #mv
Reply-To: → #rt

Ad 2.

U hromadných emailových zpráv, které jsou určeny pro více příjemců (desítky až stovky), je vhodné projít jejich seznam a zkontorlovat, zda se některé názvy domén neopakují. Názvy, které se vyskytují více než jednou, nahrazuji písmenem vyjadřujícím jejich pořadí v seznamu názvů domén, pro přehlednost seřazeném podle četnosti výskytu od nejvyššího.

Příklad:

aa@seznam.cz, bb@seznam.cz, cc@volny.cz, dd@seznam.cz, ee@volny.cz, ff@centrum.cz

bude nahrazeno tímto krazším řetězcem:

aa@a, bb@a, cc@b, dd@a, ee@b, ff@c

Ad 3.

Položka Date: obsahuje celkem dlouhý řetězec určující přesný čas odeslání. Vypadá např. takto: Thu, 03 Mar 2011 10:44:09 +0100 (CET). I když to na první pohled není zcela zřejmé, tento časový údaj je složen z celých 37 znaků! C# nám umožňuje tento časový údaj převést na celé číslo, které je, i když obvykle vysoké, asi o polovinu kratší než výše uvedený zápis. Toto číslo reprezentuje časový údaj v sekundách uplynulých od půlnoci 1.1.0001. Tento údaj by teoreticky bylo možné ještě zkrátit, kdybychom jej nahradili např. počtem sekund uplynulých od půlnoci 1.1.1950, protože je celkem vysoká šance, že příliš často nebudem pracovat s emailovými zprávami odeslanými před tímto datem. Bližší informace o této konverzi si uvedeme v kapitole zabývající se samotnou implementací. Zde si dovolím pouze jeden příklad pro představu:

Wed, 24 Feb 2010 09:42:02 → 63402601322

Komu se tento údaj jeví na první pohled jako příliš nízký, si může snadno ověřit jeho pravdivost např. rychlým porovnáním s údajem, kolik sekund uběhlo do začátku roku 2011:

60	***60**	***24**	***365**	***2011**	**=**	**63 418 896 000**
sekundy	minuty	hodiny	dny	roky		

Ad 4.

Není zcela jednoznačné, zda tento bod patří ještě ke zpracování hlavičky, nebo se již jedná o zpracování těla, proto jej uvádím až jako poslední. Boundary se uvádí v poli hlavičky Content-Type, a jak můžeme vidět na níže uvedeném příkladu, odděluje jednotlivé části zprávy a také je uvedeno na začátku a na konci těla zprávy. Všimněte si, že se rovněž jedná o celkem dlouhý řetězec, který ale pro naši potřebu můžeme, s klidným srdcem, nahradit například výrazně kratším řetězcem „#b". Když si uvědomíme, kolikrát až může být tento oddělovač uveden ve zprávách, které obsahují více částí, zjistíme, že se taktéž jedná o nezanedbatelnou úsporu.

Ukázka použití „boundary":

```
Content-Type: multipart/alternative; boundary=001636416dd1628528048054e218
X-Smtpd: 1.1.10@14229

--001636416dd1628528048054e218
Content-Type: text/plain; charset=ISO-8859-1

Me too.

2010/2/24 Honza N. <Honza.N@seznam.cz>

> Helou world.
--
JN

--001636416dd1628528048054e218
Content-Type: text/html; charset=ISO-8859-1
Content-Transfer-Encoding: quoted-printable

Me too.<br><br><div class=3D"gmail_quote">2010/2/24 Honza N. <span dir=3D"l=
tr">&lt;<a href=3D"mailto:Honza.N@seznam.cz">Honza.N@seznam.cz</a>&gt;</span><b=
r><blockquote class=3D"gmail_quote" style=3D"border-left: 1px solid rgb(204=
, 204, 204); margin: 0pt 0pt 0pt 0.8ex; padding-left: 1ex;">
Helou world.<br>
</blockquote></div><br clear=3D"all">-- <br>JN<br>

--001636416dd1628528048054e218--
```

4.2 Zpracování těla a přílohy

Jak již bylo uvedeno na začátku této kapitoly, zpracováním těla a příloh emailových zpráv se v této práci nebudeme zabývat příliš obšírně. Jelikož tělo zprávy je složeno výhradně z textu, případně z text ve formátu HTML, není potřeba vyvýjet žádný speciální kompresní algoritmus určený výhradně pro kompresi těl emailových zpráv. Ve své práci tedy k tomuto účelu používám, kromě níže zmíněných metod, zip, který je dnes velmi populární a hojně rozšířený. Zip používám stejně tak i ke kompresi příloh zpráv, protože, jak jsme si řekli v úvodu kapitoly, opravdu není možné v rozsahu této práce implementovat různé kompresní algoritmy pro všechny možné typy příloh.

Jednou věcí, o které bych se zde zmínil, je **komprese citací** použitých v těle zprávy. Nebo spíš bych měl napsat odstraňování redundatních částí textu, které pouze opakují jinou zprávu nebo její část. Protože často se stává, že píšeme odpověď na odpověď odpovědi atd. a nakonec naše zpáva může ve svém těle obsahovat citace předchozích deseti zpráv, které ale ovšem taktéž archivujeme v naší kolekci. Uvádím to zde ale pouze jako možnost dalšího budoucího vylepšení mého frameworku, protože dohledávání originálních zpráv ke každé použité citaci by bylo časově dosti náročné.

Další možností, kterou bych zde rád uvedl a kterou již můj framework nabízí, je **odstranění části těla ve formátu HTML** a zanechání pouze plain-textu, nebo naopak. Mnoho dnešních emailových klientů nabízí formátování těl zpráv pomocí použití HTML tagů. V takovémto případě nastává v těle zprávy ke zdvojení napsaeného textu – jednou je zde uveden pouze jako čistý neformátovaný plain-text a následně jako HTML doplněný o patřičné tagy. Jak jsme si již dříve uvedli, je to z toho důvodu, že ne všichni emailoví klienti podporují formát HTML. Je to tedy další možnost komprese, kterou není možné použít, kvůli její ztrátovosti, ve všech případech. Hodí se použít ji pouze tehdy, když víme, že nám v budoucnu bude stačit pouze plain-textová verze, nebo když máme jistotu, že budeme vždy používat už jen klienty, co si s HTML bez problémů poradí.

V příkladu uvedeném na předchozí stránce (ukázka „boundary“) můžeme také vidět názorné použití formátu HTML. Zároveň si uvědomme, co ze 4 slov, datumu a jedné emailové adresy v prostém plain-textu, dokáže HTML „vyrobit“ za obsáhlý odstavec a kolik místa bysme tedy byli schopni uspořit, kdybychom si vystačili při archivování jen s plain-textem.

Jakých výsledků jsme schopni těmito různými způsoby komprese dosáhnout si přehledně a podrobně uvedeme v jedné ze závěrečných kapitol.

Vraťme se ale ještě ke **zpracování příloh**. Protože každá příloha emailové zprávy má svou vlastní hlavičku, zpracovávám i tyto hlavičky podobně jako hlavičku celé zprávy. V tomto případě opět nahrazuji dlouhá jména polí hlavičky jejich daleko kratšími ekvivalenty. Zde je ukázka:

Content-Type: → #ct
Content-Disposition: → #cd
Content-Transfer-Encoding: → #ce
X-Attachment-Id: → #ai

Samotná těla příloh jsou v emailových zprávách většinou kódovány pomocí base64, které binární data převádí na tisknutelné ASCII znaky. Base64 má pro nás jednu nevýhodnou vlastnost, a to, že délku původního řetězce navýší přibližně o třetinu. To je zapříčiněno tím, že toto kódování převádí každé 3 bajty původních dat na 4 ASCII znaky. Proto, ještě než se provede hlavní komprimace pomocí zipu, každou přílohu oddělím od emailové zprávy do samostatného souboru a převedu z base64 zpět do binární podoby. Informace o kódování base64 čerpány ze zdroje [7].

Na závěr kapitoly malá ukázka, jak vypadá jeden odstavec přiloženého textu v base64:

```
TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG5vdCBvbmx5IGJ5IGhpcyByZWFzb24sIGJ1dCBieSB0aGlz
IHNpbmd1bGFyIHBhc3Npb24gZnJvbSBvdGhlciBhbmltYWxzLCB3aGljaCBpcyBhIGx1c3Qgb2Yg
dGhlIG1pbmQsIHRoYXQgYnkgYSBwZXJzZXZlcmFuY2Ugb2YgZGVsaWdodCBpbiB0aGUgY29udGlu
dWVkIGFuZCBpbmRlZmF0aWdhYmxlIGdlbmVyYXRpb24gb2Yga25vd2xlZGdlLCBleGNlZWRzIHRo
ZSBzaG9ydCB2ZWhlbWVuY2Ugb2YgYW55IGNhcm5hbCBwbGVhc3VyZS4=
```

Dekódování zpět do čitelného textu nám dá tento výsledek:

Man is distinguished, not only by his reason, but by this singular passion from other animals, which is a lust of the mind, that by a perseverance of delight in the continued and indefatigable generation of knowledge, exceeds the short vehemence of any carnal pleasure.

Jak vidíme, výsledný text je opravdu přibližně o třetinu kratší, než zakódovaný pomocí base64.

5. Popis požadavků specifických pro kompresi emailových zpráv

Na kompresi emailových zpráv můžeme nahlížet v mnoha ohledech jako na kompresi každých jiných dat. Jak po přečtení předchozích kapitol jistě tušíte, můžeme provozovat jak kompresi ztrátovou, tak kompresi bezeztrátovou. Kompresi ztrátovou dokonce v několika stupních, podle aktuální potřeby.

Asi nejdůležitějším specifikem komprese emailových zpráv je to, že, narozdíl od většiny ostatních formátů, má takováto zpráva tři rozdílné části, ke kterým je nutno přistupovat odlišně.

Jak jsme si již dříve uvedli, **komprese hlavičky** nám na základě jasně stanovených standartů, kterými se při svém sestavování řídí, nabízí více různých možností nahrazení standartních nebo často se opakujících řetězců jejich kratšíi ekvivalenty. Případně některé části hlavičky můžeme zcela vypustit, uznáme-li je za nepotřebné pro budoucí činnost. A až následně k celé hlavičce budeme přistupovat jako ke stanrartnímu textu.

U **těla** zprávy nastává odlišná situace – rovněž se sice jedná o standartní text, ale již pro něj neplatí žádné zákonitosti, již zde nenajdeme žádné povinné struktury apod. Zbývá nám tedy opět již dříve zmiňovaná možnost volby, zda ponechat všechny části (např. plain-text i HTML) nebo si pro budoucí použití zvolíme pouze některé z nich a ostatních se v rámci ztrátové komprese zbavíme nadobro.

Klíčovou částí komprese emailových zpráv bude nejspíše vždy **komprese příloh**, protože ty v drtivé většině případů tvoří právě tu nejobjemnější část zprávy. Problémem zde je fakt, že jako příloha může být soubor v jakémkoliv formátu. Buď tedy můžeme používat speciální algoritmy pro každý formát zvlášť, nebo jak tomu je v našem případě, budeme k přílohám přistupovat jednotně, za cenu horšího výsledku komprese.

Zajímavou možností, použitelnou např. u serverů poskytujících poštovní schránky, by bylo určovat stupeň komprese podle stáří zprávy. Zatímco u mladších zpráv by se používat lehčí typ komprese, aby v případě potřeby byly dostupné v reálném čase, u zpráv staších, kde je možnost jejich otevření minimální, by bylo možné použít složitějších a účinějších kompresních algoritmů, třeba i na skupiny více zpráv najednou. S tím, že následná dekomprimace by už zabrala i nějaký ten čas. To už je ale ovšem naprosto mimo rámec této práce.

6. Implementace

K implementaci mé aplikace jsem použil architektury .NET, konkrétně jazyk C#. Kompletní aplikaci včetně knihovny EmailLib jsem naprogramoval pomocí Microsoft Visual Studia 2008.

6.1 .NET Framework

Architektura .NET je soubor technologií, poskytujících množství předvytvořených řešení umožňujících vývoj a spouštění aplikací, vytvořených v různých programovacích jazycích pro Web, stolní i mobilní počítače. Nejdůležitějšími součástmi jsou společné běhové prostředí (CLR) a knihovna tříd, podobně jako u technologie Java. Knihovna tříd nabízí využití předvytvořených řešení k vytváření vlastních programů. Je dostupná pro všechny programovací jazyky určené pro architekturu .NET. Součástí této knihovny jsou např. prostředky pro tvorbu uživatelského rozhraní, pro práci s databázemi, pro vývoj webových aplikací, pro síťovou komunikaci, pro práci se soubory apod. Společné běhové prostředí se stará o spouštění a řízení programů. Programy jsou spouštěny na virtuálním stroji, což umožňuje lepší přenositelnost kódu, protože vývojář nemusí znát cílovou platformu, kde bude program provozován. Toto prostředí také poskytuje prostředky pro bezpečnost, řízení paměti a ošetřování výjimek. Programy mohou být napsány v různých programovacích jazycích a výsledný kód je přeložen do mezikódu (CIL). Tento kód je překládán do nativního kódu až ve chvíli potřeby (JIT překlad).

V současnosti existuje přes 40 jazyků s kompilátorem pro .NET, většina z nich je vytvořena dodavateli třetích stran. První verze tohoto Frameworku vyšla v roce 2002. Verze 1.1, která znamenala první větší upgrade, byla použita poprvé jako součást operačního systému. Dnes je asi nejrozšířenější verze s označením 2.0.

Většina zde použitých informací byla čerpána ze zdroje [6].

6.2 Aplikace

Uživatelské rozhraní aplikace je potom tvořeno komponentami WindowsForms. Většinu aplikační části obstarávají formuláře/třídy frmCompression a frmDecompression. Samotnou kompresi potom zajišťuje knihovna EmailLib a její třída Compression s pomocí tříd Parser a Message.

6.2.1 Funkce aplikace

Celý proces se zahájí výběrem emailových zpráv určených pro kompresi. Tuto akci ulehčuje user control ucDirectoryTree, který slouží k procházení adresářového stromu. Z vybraných emailů je potom vytvořen seznam, který je následně postupně zpracováván. Jednotlivé emaily jsou po jednom načítány a dle zvoleného typu komprese dále parsrovány na jednotlivé části a komprimovány. Nejdříve hlavička, potom tělo a na závěr příloha, která je vyčleněna do samostatného souboru nebo je odstraněna úplně. Takto upravený email je potom (společně s případnou přílohou) přidán do archivu a pokračuje se dalším až do vyčepání seznamu. Dekomprimace se provádí obráceným postupem.

6.3 Implementace parseru pro emailové zprávy

Samotný parser emailových zpráv je překvapivě ukrytý ve třídě Parser. Ta obsahuje jednu hlavní veřejnou metodu Parse, která na svém vstupu očekává emailovou zprávu ve formě řetězce. Následně tuto zprávu rozdělí pomocí množství privátních metod, z nichž si několik uvedeme pro zajímavost níže. Těmito částmi (jednotlivé hlavičky zprávy, tělo, přílohy) potom naplní nově vzniklý objekt Message, který na závěr vrátí a se kterým dále pracuje třída pro kompresi nebo např. formulář pro zobrazení vybrané zprávy.

Zde je tedy slibovaná ukázka privátních metod třídy Parser:

1. Metoda GetDate pro získání položky „Date“ z hlavičky zprávy.

```
private String GetDate (String szMessage)
{
   szMessage = szMessage.Remove (0, (szMessage.IndexOf ("\r\nDate:") + 7));
   szMessage = szMessage.Remove (szMessage.IndexOf ('\r'), (szMessage.Length -
               szMessage.IndexOf ('\r')));

   return szMessage;
}
```

2. Metoda GetMultipart pro zjištění, zda zpráva je “multipart” (z hlavičky “Content-Type”) a pokud ano, tak získání hodnoty jejího “boundary” pro další použití. Jak vidíte, hlavní náplní těchto metod je postupně ořezat vstupní řetězec pomocí metody Remove třídy String, až se dopracujeme k požadované části naší zprávy.

```
private Boolean GetMultipart (String szMessage)
{
   String szMessOrig = szMessage;
   szMessage = szMessage.Remove (0, (szMessage.IndexOf ("\r\nContent-Type:"+6));
   szMessage = szMessage.Remove (szMessage.IndexOf ("\r\n"), (szMessage.Length -
               szMessage.IndexOf ("\r\n")));

   bIsMultipart = szMessage.Contains ("multipart");

   if (bIsMultipart)
   {
      szMessage = szMessOrig.Remove (0, (szMessOrig.IndexOf ("boundary=")));
      szMessage = szMessage.Remove (szMessage.IndexOf ("\r\n"),
                  (szMessage.Length - szMessage.IndexOf ("\r\n")));

      szBoundary = szMessage.Remove (0,szMessage.IndexOf("=") + 1).Trim('\"');
   }

   return bIsMultipart;
}
```

6.4 Implementace frameworku

Na tomto místě bych se rád trochu zmínil o implementaci frameworku. Jelikož hlavním úkolem tohoto frameworku je komprese a dekomprese emailových zpráv, zaměřím se zde především na tuto část. Koho by více zajímal "obal" tohoto jádra, toho odkážu na uživatelskou a programátorskou příručku, popř. na samotné zdrojové kódy, protože této méně zajímavé části se zde věnovat nebudeme.

6.4.1 Nahrazení názvů polí hlavičky

Začneme metodou nahrazující názvy jednotlivých položek hlavičky zprávy. Vidíme, že náplní této metody je opravdu pouze jednoduché postupné nahrazování názvů jejich kratšímy ekvivalenty. Využívám k tomu metodu Replace třídy String, která jako vstupní parametry bere původní řetězec a jeho nový tvar, kterým má být v celém textu nahrazen. Jedinou zvláštností zde je to, že některé položky jsou uvedeny ve více tvarech (zde např. Message-Id). Je to z toho důvodu, že různí poštovní klienti tuto položku zapisují jinak a našim cílem je tyto různé zápisy sjednotit. Metoda je v originále samozřejmě delší, ale pro ukázku nám myslím bude stačit prvních několik řádků. Dekompresní metodu již uvádět nebudu protože je totožná s touto až na opačný směr nahrazování řetězců.

```
private static String RepleaceHeaders(String szText)
{
   szText = szText.Replace("Received:", "#re");
   szText = szText.Replace("Date:", "#d");
   szText = szText.Replace("Message-ID:", "#mi");
   szText = szText.Replace("Message-Id:", "#mi");
   ...

   return szText;
}
```

6.4.2 Nahrazení názvů domén v seznamu adresátů

Tuto část zde popíši pouze ústně, protože kód je poněkud obsáhlejší a uvedený zde na několik stránek by jistě ztratil svou přehlednost. Proto se pokusím zde uvést alespoň stručný postup, který myslím pro nikoho nebude nijak překvapivý.

1. Nejdříve seznam adresátů rozdělím metodou Split třídy String na jednotlivé adresy a vytvořím z nich jednorozměrné pole.

2. Toto pole adres následně postupně procházím a z každé adresy získám název domény, ke které náleží. Tento název domény poté uložím do nového pole domén, které obsahuje název domény a četnost jejího výskytu.

3. Po projití celého seznamu adres seřadím seznam domén podle četnosti jejich výskytu a přiřadím každé jedno písmeno abecedy odpovídající jejich pořadí. (Tento seznam domén + písmena jim přiřazené přiložím nakonec celé emailové zprávy pro zpětné nahrazení při dekompresi.)

4. No a na závěr opět pomocí metody Replace nahradím názvy domén jim přiřazenými písmeny.

Názorný příklad takovéto záměny byl uveden ve 4. kapitole.

6.4.3 Nahrazení datumů

Zde se dostáváme k zajímavější části a to jest nahrazení datumových položek. Jak vidíme, jádrem níže uvedené metody je pouze jeden řádek, ten ale spojuje dohromady několik kroků. Pojdmě se na ně podívat podrobněji. Nejdříve z datumu uvedeného v textové podobě získáme objekt DateTime. Dále nás bude zajímat vlastnost tohoto objektu s názvem Ticks, o které jsem se již zmiňovali ve 4. kapitole. Nakonec toto číslo vydělíme 10 000 000 a získáme výsledný počet sekund, který metoda vrací. Toto číslo již zpracuje jiná metoda, která jej převede do textové podoby metodou ToString a metodou Replace třídy String jej nahradí za všechny výskyty původního datumu v celé zprávě. Ještě si můžeme povšimnout, že při výskytu chyby při převodu metoda vrátí 0, která oznamuje, že k žádnému nahrazení v tomto případě dojít nemá.

```
static private Int64 DateToLong (String szDate)
{
   try
   {
      return Convert.ToDateTime (szDate).Ticks / 10000000;
   }
   catch (FormatException)
   {
      return 0;
   }
}
```

Opačný směr užívaný při dekomprimaci vypadá takto. Vezmeme číslo označující počet sekund a vynásobíme jej 10 000 000. To nám dá hodnotu Ticks, kterou můžeme nyní použít pro vytvoření nového objektu DateTime. Když na tento objekt následně použijeme metodu ToString s parametrem "R", získáme naše původní datum, přesně v tom formátu, v jakém bylo před komprimací.

Tento řetězec vrátime metodě, která se již postará o nahrazení datumů v číselném tvaru za tento původní tvar v celé naší zpracovávané emailové zprávě.

```
static private String LongToDate (Int64 i64Date)
{
   return new DateTime(i64Date * 10000000) .ToString ("R");
}
```

6.4.4 Dekódování přílohy

Poslední částí, kterou bych se zde rád zabýval, je dekódování a zpětné kódování příloh. Nejdříve se podíváme na dekódování. Přílohy jsou k emailové zprávě připojeny zakódovány pomocí base64. Z důvodu zmenšení koncové velikosti je před archivací dekóduji. Na starosti to má níže uvedená metoda, která vstupní řetězec dekóduje na pole bytů a toto pole potom zapíše do nového souboru, který se nakonec přejmenuje a připojí k archívu.

```
private static void DecodeFrom64(String encodedData)
{
   FileStream fs = new FileStream(@"attachment", FileMode.OpenOrCreate);

   byte[] encodedDataAsBytes = Convert.FromBase64String(encodedData);

   fs.Write(encodedDataAsBytes, 0, encodedDataAsBytes.Length);
   fs.Close();
}
```

Při dekomprimaci takto vytvořený soubor s přílohou otevřu a načtu do pole bytů. Toto pole potom pomocí metody ToBase64String třídy Convert převedu zpět na řetězec zakódovaný v base64. Výsledný řetězec na závěr připojím zpátky ke zprávě, ke které náleží a od které byl při komprimaci oddělen.

```
private static String EncodeTo64(Byte[] toEncode)
{
   return System.Convert.ToBase64String(toEncode);
}
```

7. Zhodnocení efektivity komprese pomocí frameworku

V první níže uvedené tabulce můžeme vyčíst, jak si naše aplikace poradila s testovacím vzorkem obsahujícím několik jednoduchých emailů, několik emailů s vícenásobným příjemcem, několik s HTML tělem a několik emailů s přílohami různých typů.

Jak vidíme, tak výsledná účinnost komprese se pohybuje okolo 60% podle zvoleného typu komprese. Kromě poslední ztrátové komprese, kdy odebíráme i přílohy. Zde je výsledek opravdu zajímavý. Ale praktická použitelnost tohoto algoritmu je velmi diskutabilní. Jak již bylo zmíněno v jedné z předchozích kapitol – tento typ je vhodný spíše jen pro soukromé účely, kdy víme, že s emaily již nebudeme dále prakticky manipulovat a nevadí nám, že při komprimaci přijdeme o přílohy (např. námi odeslané emaily). Nebo ještě tento typ komprese může být vhodný pro společnosti, kde je nutné uchovávat objemnou emailovou komunikaci po dobu několika let a základní údaje z hlavičky + tělo zprávy jsou pro archivaci plně postačující. Úspora diskového prostoru v těchto případech je markantní a může se pohybovat až v řádu terabajtů.

Popis	**Velikost**	**Velikost na disku**	**V procentech**
Originální data	2,61 MB	2,64 MB	100%
Po kompresi zipem	1,76 MB	1,76 MB	67,4%
Po obyčejné kompresi	1,67 MB	1.68 MB	64,00%
Po ztrátové kompresi	1,46 MB	1,46 MB	55,90%
Po ztrátové kompresi (bez příloh)	10,7 kB	12,0 kB	0,40%

Tabulka 1 – Účinnost různých typů komprese na smíšenou kolekci emailů.

Popis	Velikost	Velikost na disku	V procentech
Originální data	70,8 kB	112 kB	100%
Po kompresi zipem	31,7 kB	32,0 kB	44,8%
Po obyčejné kompresi	31,1 kB	32,0 kB	43,9%
Po ztrátové kompresi	21,5 kB	24,0 kB	30,4%

Tabulka 2 – Účinnost různých typů komprese na kolekci emailů bez přílohy.

Na druhé tabulce vidíme, že výsledek komprese první kolekce byl negativně ovlivněn přítomností příloh, které je tímto způsobem možno zkomprimovat přibližně na velikost výsledných 60%. Ve druhém případě se úspěšnost komprese pohybuje lehce nad 40%, což je výsledek výrazně lepší. Co je naopak výrazně horší je výsledek ztrátové komprese ovlivněný nepřítomností příloh, které by bylo možné odstranit. Vidíme, že odstraněním nadbytečných částí zpráv jsme v tomto případě ušetřili okolo pouhých 14% oproti kompresi bezztrátové.

Na závěr ještě jedna poznámka - obecně můžeme říci, že u kratších emailových zpráv tohoto typu, kde hlavička tvoří větší část emailu, bude účinnost komprese mírně vyšší než u zpráv dlouhých.

8. Závěr

Cílem této práce bylo napsat framework pro kompresi emailových kolekcí. Základem tohoto frameworku je parser pro emailové zprávy, který výsledná aplikace používá jednak pro zobrazení emailových zpráv uživateli, ale především pro samotnou kompresi. V této práci jsem se především zaměřil na kompresi hlavičky emailových zpráv, kdy dochází k nahrazování často používaných dlouhých řetězců jejich kratšími ekvivalenty. Neméně podstatnou částí komprese je ale i převod příloh z kódování base64 zpět do původní binární podoby, která je přibližně o čtvrtinu méně prostorově náročná.

Během testování mé aplikace se ukázalo, že i tyto jednoduché postupy jsou schopny snížit velikost emailových kolekcí na přibližně 60% jejich původní velikosti. A v extrémních případech, kdy nám záleží pouze na zachování těch nejelementárnějších informací, které jsou ale stále ještě použitelné, se můžeme přiblížit až k hranici 0,5% z původní velikosti. Ovšem za předpokladu ztráty veškerých příloh a většiny položek hlavičky těchto emailových zpráv.

Při vývoji této aplikace jsem se do podrobna seznámil se strukturou emailových zpráv a se způsobem jejich parsrování pro další použití. Dalším zajímavým poznatkem pro mě byla práce s přílohami a jejich kódováním.

8.1 Možná vylepšení

Jak již bylo několikrát zmíněno v průběho této práce, nebylo v mých silách naimplementovat individuální algoritmus pro každý z možných typů příloh. Jistým zlepšením by tedy bylo použití různých kompresních algoritmů alespoň pro nejběžneji používané datové formáty.

Stejně tak by přinesla zefektivnění implementace speciálního algoritmu pro komprimaci prostého a formátovaného HTML textu těl emailů nebo zmíněná komprimace redundantních citací.

Zajimavá by jistě byla i proměnlivá úroveň komprese podle stáří emailové zprávy.

Poslední možným vylepšením, které mě při tvorbě aplikace napadlo, by byla možnost individuální volby každého uživatele, které atributy při ztrátové kompresi zachovat a které je možno při kompresi odstranit.

9. Literatura

[1] **RFC 5322.** [Online] http://tools.ietf.org/html/rfc5322, 28.3.2011

[2] **RFC 2045 - 2049.** [Online] http://tools.ietf.org/html/rfc2045 - 2049, 28.3.2011

[3] **Data Compression: The Complete Reference**. D. Salomon, Springer, 2004, ISBN 978-0387406978

[4] **RFC 822.** J. Peterka. [Online] http://www.earchiv.cz/a94/a435c110.php3, 28.3.2011

[5] **Wikipedia.** [Online] http://en.wikipedia.org/wiki/Email, 28.3.2011

[6] **Microsoft .NET compact framework.** A. Wigley and S. Wheelwright. Redmond : Microsoft Press, 2002.

[7] **Wikipedia.** [Online] http://en.wikipedia.org/wiki/Base64, 28.4.2011

A Uživatelská příručka

Systémové požadavky

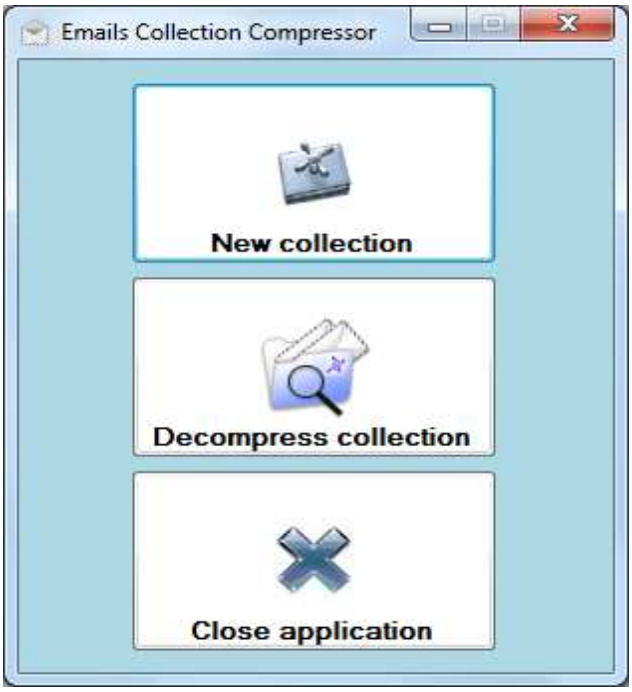
K provozu této aplikace je zapotřebí mít v systému nainstalován .NET Framework 2.0.

Instalace a konfigurace

Aplikace nepotřebuje instalaci, stačí zkopírovat adresář s aplikací na vybrané místo. Aplikace není nutné jakkoliv konfigurovat, po svém startu by měla být již plně funkční.

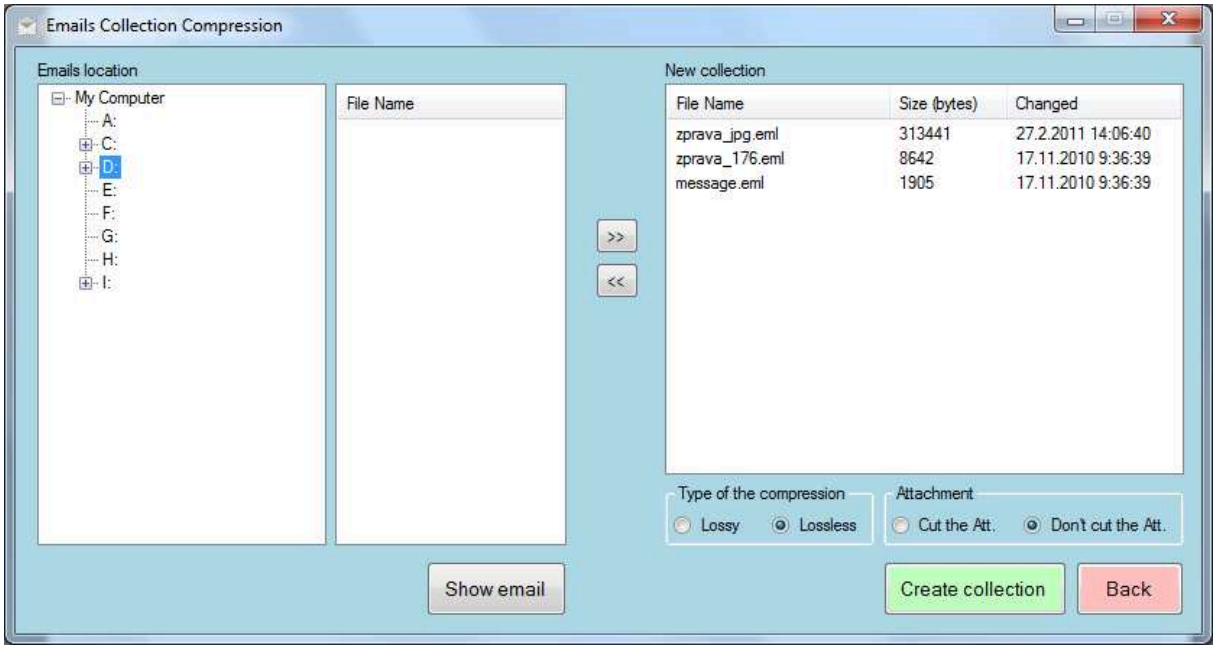
Práce s aplikací

Aplikace se spouští souborem EmailCollectionCompressor.exe. Po startu se objeví okno, kde si uživatel zvolí, kterou akci chce provést – zda vytvořit novou kolekci, nebo rozbalit již existující kolekci. Poslední volbou se práce s aplikací ukončí.

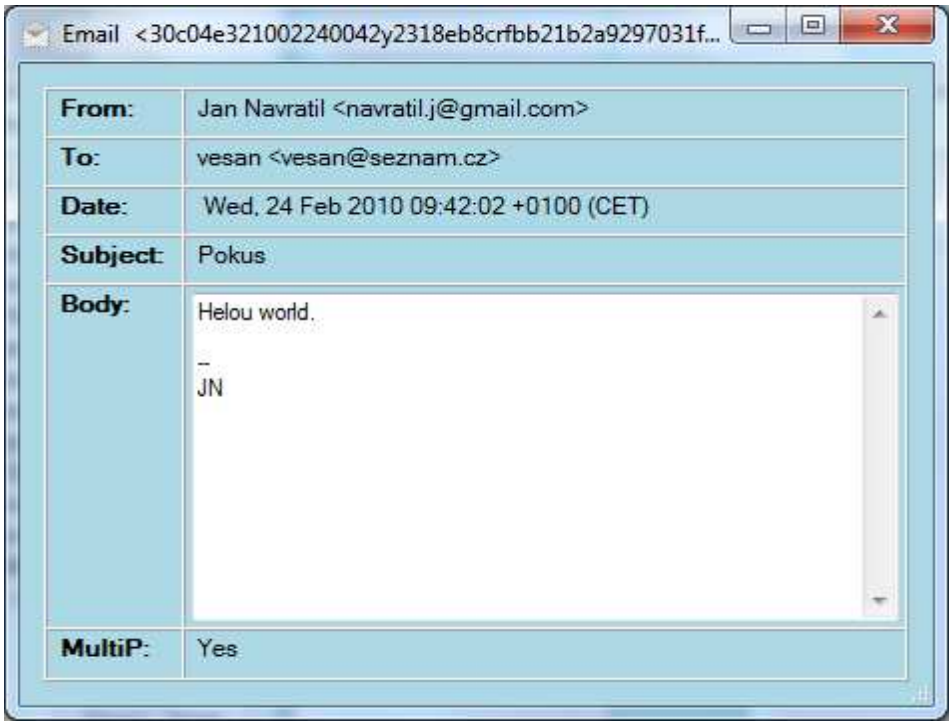


Obrázek 2 – Hlavní menu aplikace.

Po volbě vytvoření nové kolekce se uživateli otevře toto okno. V levé části nadepsané „Emails location“ uživatel vybere adresář, ve kterém se nacházejí emaily určené k archivaci. Obsah vybraného adresáře se zobrazí ve vedlejším sloupci. Pomocí šipky doprava může uživatel vybrat, které emaily si přeje archivovat. Šipkou doleva vybrané emaily z výběru odstraní. Až jsou v seznamu napravo všechny vybrané emaily, uživatel pomocí přepínačů ve spodní části zvolí, zda si přeje kompresi ztrátovou či bezztrátovou a jestli si přeje přílohy odebrat nebo zachovat. Pak už stačí zmáčknout tlačítko „Create collection“ pro vytvoření kolekce nebo tlačítko „Back“ pro zavření okna. Poslední tlačítko „Show email“ slouží k zobrazení vybraného emailu ze seznamu v levé části.

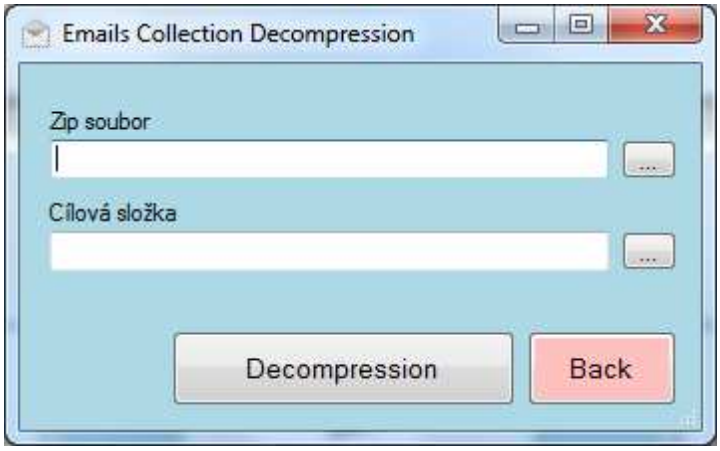


Obrázek 3 – Okno pro vytvoření nové kolekce.



Obrázek 4 – Okno s náhledem emailové zprávy.

Poslední obrázek zobrazuje okno pro dekomprimaci archivu emailů. Do prvního pole zadá uživatel pomocí dialogu cestu k archívu, který si přeje rozbalit. Do pole druhého potom zadá opět pomocí dialogu cílovou složku, kam si přeje archiv rozbalit. Akci potvrdí tlačítkem „Decompression“ nebo okno zavře tlačítkem „Back“.



Obrázek 5 – Formulář pro rozbalení kolekce.

B Programátorská dokumentace

Program je implementován jako Windows Application v C#. Skládá se z projektu EmailCollectionCompressor a z knihovny EmailLib. Veškeré zdrojové soubory jsou na přiloženém CD v adresáři Zdrojové kódy.

Projekt EmailCollectionCompressor obsahuje především 4 formuláře z nichž se moje aplikace skládá. Nejdůležitějším a nejobsáhlejším je samozřejmě formulář frmCompression, který slouží k vytváření nových komprimovaných kolekcí. Dále se v tomto projektu např. nachází komponenta ucDirectoryTree, která ve výše zmíněném formuláři slouží k procházení adresářového stromu.

Knihovna EmailLib obsahuje 3 třídy. Třída Message reprezentuje svojí strukturou emailovou zprávu. Třída Parser slouží k naplnění této struktury ze zrdojové zprávy. Ta se dále použije buď pro zobrazení zprávy pomocí frmMessage, nebo se použije dále při komprimaci prováděné třídou Compression. Tato třída má na starosti vše, co se týká komprimace emailové zprávy – zajišťuje jak ztrátovou, tak bezztrátovou kompresi, a navíc se také také používá při dekomprimaci archivů. Nejdříve jsem zde chtěl prezentovat některé zajímavé části kódu, ale došel jsem k závěru, že nemá smysl zde vystavit pouze fragmenty celé této obsáhlé třídy vytržené z kontextu. Pro zájemce je tedy její zdrojový kód k nahlédnutí na přiloženém CD, její metody jsou jasně a srozumitelně pojmenovány, takže by pro nikoho neměl být žádný problém najít požadovanou část kódu.

C Obsah CD

Adresář	**Popis**
/Text	Tento dokument v elektronické podobě.
/Source	Adresář s kompletními zdrojovými kódy (Visual Studio Solution).
/Install	Spustitelná aplikace s potřebnými soubory.